TEC | Tecnológico de Costa Rica

Tarea Número 1 (Capítulo 1)

Introducción y Taller de programación
I-Semestre 2022

Estudiante Marco Rodríguez
Estudiante Maximilian Latysh

Prof. Edgar Rojas

24 de febrero de 2022

1. 1.8.1. Problemas de Funciones

Sobre conjuntos infinitos:

1.1. Una función ni sobreyectiva ni inyectiva

$f : \mathbb{R} \to \mathbb{R}, f(x) = (x+3)(x+1)(x-1)(x-3)$

Su ámbito es $[16, +\infty[$ y su codominio es $\mathbb{R}$, por lo tanto $f(x)$ no es sobreyectiva. Por otra parte, la estructura de la función crea repeticiones de valores en el ámbito, lo que hace $f(x)$ no inyectiva.

$\exists x \in D_f[f(x) \notin C_f] \wedge \exists a, b \in D_f[a \neq b / f(a) = f(b)]$

1.2. Una función sobreyectiva pero no inyectiva

$g : \mathbb{R} \to \mathbb{R}, g(x) = x(x+4)(x+2)(x-2)(x-4)$

Su ámbito es $\mathbb{R}$ y su codominio es $\mathbb{R}$, por lo tanto $g(x)$ es sobreyectiva. Por otra parte, la estructura de la función crea repeticiones de valores en el ámbito, lo que hace $g(x)$ no inyectiva.

$\forall x \in D_g[g(x) \in C_g] \wedge \exists a, b \in D_g[a \neq b / g(a) = g(b)]$

1.3. Una función no sobreyectiva pero sí inyectiva

$h : \mathbb{R} \to \mathbb{R}, h(x) = 2^x$

Su ámbito es $]0, +\infty[$ y su codominio es $\mathbb{R}$, por lo tanto $h(x)$ no es sobreyectiva. Por otra parte, la estructura de la función no crea repeticiones de valores en el ámbito, lo que hace $h(x)$ inyectiva.

$\exists x \in D_h[h(x) \notin C_h] \wedge \forall a, b \in D_h[a \neq b / h(a) \neq h(b)]$

1.4. Una función biyectiva

$j : \mathbb{R} \to \mathbb{R}, j(x) = x + sin(x)$

Su ámbito es $\mathbb{R}$ y su codominio es $\mathbb{R}$, por lo tanto $j(x)$ es sobreyectiva. Por otra parte, la estructura de la función no crea repeticiones de valores en el ámbito, lo que hace $j(x)$ inyectiva. Eso podemos verificar conociendo el hecho de que el máximo valor de $j'(x) = 1$. Por lo tanto, el cambio que se provoca en el término $sin(x)$ nunca supera al cambio provocado por el término x.

$\forall x \in D_j[j(x) \in C_j] \wedge \forall a, b \in D_j[a \neq b / j(a) \neq j(b)]$

2. 1.8.3. Criba de Eratóstenes

Es un algoritmo antiguo para encontrar números primos en un rango de 1 a x por lo tanto que $x > 1$. El algoritmo consiste en sacrificar espacio computacional para ganar rapidez. Consiste en encontrar un número primo y marcar todos sus múltiplos hasta x. Se busca el siguiente número que no esté marcado (este será primo). De nuevo, marcamos todos sus múltiplos. Así se repite el algoritmo hasta llegar a x [1].

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40	41	42	43	44	45
46	47	48	49	50	51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70	71	72	73	74	75
76	77	78	79	80	81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100	101	102	103	104	105
106	107	108	109	110	111	112	113	114	115	116	117	118	119	120
121	122	123	124	125	126	127	128	129	130	131	132	133	134	135
136	137	138	139	140	141	142	143	144	145	146	147	148	149	150
151	152	153	154	155	156	157	158	159	160	161	162	163	164	165
166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
181	182	183	184	185	186	187	188	189	190	191	192	193	194	195
196	197	198	199	200	201	202	203	204	205	206	207	208	209	210
211	212	213	214	215	216	217	218	219	220	221	222	223	224	225
226	227	228	229	230	231	232	233	234	235	236	237	238	239	240

Figura 1: Todos los números primos son marcados con el anaranjado (#e83f18) para que sea más fácil de verlos.

Listing 1: Algoritmo de la Criba de Eratóstenes

```
def main(limite):
   marcados = []
   for i in range(2,limite):
      if i not in marcados:
         print(i)
         for j in range(limite//i):
            if (j+1)*i not in marcados:
               marcados.append((j+1)*i)

if __name__ == "__main__":
   limite = int(input())
   main()
```

Listing 2: Resultado del Algoritmo de la Criba de Eratóstenes

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53,
59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109,
113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173,
```

```
179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239]
```

3. 1.8.4. Los cuatro cuadrados

3.1. 1

$1 = 1^2 + 0^2 + 0^2 + 0^2$
$= 1 + 0 + 0 + 0$

Tomamos la raíz de 1 y como el resultado es un número natural podemos saber que 1^2 es la respuesta.

3.2. 2

$2 = 1^2 + 1^2 + 0^2 + 0^2$
$= 1 + 1 + 0 + 0$

Tomamos la raíz de 2 pero como el resultado no es un número natural lo tenemos que redondear hasta el número natural más cercano que sea menor que este. Después, restamos el cuadrado de este resultado redondeado al valor original y lo agregamos a la respuesta. Seguidamente, repetimos todo el proceso un máximo de tres veces. Si logramos encontrar los valores que satisfacen la condición de que su suma sea igual al valor original, podemos concluir el proceso como ya tenemos la respuesta.

3.3. 3

$3 = 1^2 + 1^2 + 1^2 + 0^2$
$= 1 + 1 + 1 + 0$

Tomamos la raíz de 3 pero como el resultado no es un número natural lo tenemos que redondear hasta el número natural más cercano que sea menor que este. Después, restamos el cuadrado de este resultado redondeado al valor original y lo agregamos a la respuesta. Seguidamente, repetimos todo el proceso un máximo de tres veces. Si logramos encontrar los valores que satisfacen la condición de que su suma sea igual al valor original, podemos concluir el proceso como ya tenemos la respuesta.

3.4. 16

$16 = 4^2 + 0^2 + 0^2 + 0^2$
$= 16 + 0 + 0 + 0$

Tomamos la raíz de 16 y como el resultado es un número natural podemos saber que 4^2 es la respuesta.

Otra combinación:
$16 = 2^2 + 2^2 + 2^2 + 2^2$
$= 4 + 4 + 4 + 4$

3.5. 32

$32 = 4^2 + 4^2 + 0^2 + 0^2$
$= 16 + 16 + 0 + 0$

Tomamos la raíz de 32 pero como el resultado no es un número natural lo tenemos que redondear hasta el número natural más cercano que sea menor que este. Después, restamos el cuadrado de este resultado redondeado al valor original y lo agregamos a la respuesta. Seguidamente, repetimos todo el proceso un máximo de tres veces. Si logramos encontrar los valores que satisfacen la condición de que su suma sea igual al valor original, podemos concluir el proceso como ya tenemos la respuesta; si no, sabemos que debemos borrar el resultado y excluir el primer valor que habíamos encontrado para nuestra respuesta (en este caso 5^2). Desde ahí, repetimos todo el proceso pero incorporando la exclusión previa. Así, el algoritmo llega a seleccionar 4^2 en lugar de 5^2, lo que nos permite llegar a la respuesta correcta.

3.6. 100

$100 = 10^2 + 0^2 + 0^2 + 1^2$
$= 49 + 25 + 25 + 1$

Tomamos la raíz de 100 y como el resultado es un número natural podemos saber que 10^2 es la respuesta.

Otras combinaciones (se sacan con el mismo algoritmo pero excluyendo ciertos valores):

- $100 = 9^2 + 3^2 + 3^2 + 1^2 = 81 + 9 + 9 + 1$
- $100 = 8^2 + 6^2 + 0^2 + 0^2 = 64 + 36 + 0 + 0$
- $100 = 8^2 + 4^2 + 4^2 + 2^2 = 64 + 16 + 16 + 4$
- $100 = 7^2 + 7^2 + 1^2 + 1^2 = 49 + 49 + 1 + 1$

- $100 = 7^2 + 5^2 + 5^2 + 1^2 = 49 + 25 + 25 + 1$
- $100 = 5^2 + 5^2 + 5^2 + 5^2 = 25 + 25 + 25 + 25$

3.7. 101

$101 = 10^2 + 1^2 + 0^2 + 0^2$
$= 100 + 1 + 0 + 0$

Tomamos la raíz de 101 pero como el resultado no es un número natural lo tenemos que redondear hasta el número natural más cercano que sea menor que este. Después, restamos el cuadrado de este resultado redondeado al valor original y lo agregamos a la respuesta. Seguidamente, repetimos todo el proceso un máximo de tres veces. Si logramos encontrar los valores que satisfacen la condición de que su suma sea igual al valor original, podemos concluir el proceso como ya tenemos la respuesta

Otras combinaciones (se sacan con el mismo algoritmo pero excluyendo ciertos valores):

- $101 = 9^2 + 4^2 + 2^2 + 0^2 = 81 + 16 + 4 + 0$
- $101 = 8^2 + 6^2 + 1^2 + 0^2 = 64 + 36 + 1 + 0$
- $101 = 7^2 + 6^2 + 4^2 + 0^2 = 49 + 36 + 16$
- $101 = 6^2 + 6^2 + 5^2 + 2^2 = 36 + 36 + 25 + 4$

.

Listing 3: Algoritmo para calcular los cuatro cuadrados de cualquier número natural

```
def cuatro(x):
   #Valor original de x
   copy_x = x
   #Lista de futuras excepciones
   excpt = []
   #Lista de resultados
   l = [] #La letra L
   while True:
      cnt = 0
      while cnt < 4:
         cnt+=1
         #Magia de mate
         d = int(x**(1/2))
         fake = 0
         while d in excpt:
            fake+=1
```

```
            d = int((x-fake)**(1/2))
        #Encontramos un valor y lo agregamos a la lista de resultados
        l.append(d)
        #Hacemos la resta al valor original
        x-=d**2
    #Hacemos la suma...
    z = 0
    for i in l: z += i**2
    #Si se logra, se parra el proceso
    if z == copy_x:
        break
    #Agregamos el primer valor a las excepciones
    excpt.append(l[0])
    #Reestablecemos x
    x = copy_x
    #Nulificamos l
    l = []
return l
#print(str(copy_x)+": "+" ".join([str(i) for i in l]))
```

Nota: No, no lo robamos del internet. Desarrollamos este algoritmo nosotros solos [2].

Referencias

[1] M. P. MANZO, “Criba de eratóstenes: Si n es un número,” *Ciencia y Desarrollo en Internet*, 2004.

[2] E. R. Jiménez, *Fundamentos de Programación.* 2022.